```c
//page_replacement_FIFO
#include <stdio.h>
int main() {
    int i, j, page[100], n, capacity, frame[10];
    int pagefault = 0, frameindex = 0, frameavailable = 0, pagefound = 0;
    printf("Enter the number of pages: ");
    scanf("%d", &n);
    printf("Enter the reference string (RS): ");
    for (i = 0; i < n; i++)
        scanf("%d", &page[i]);
    printf("Enter the capacity of frames: ");
    scanf("%d", &capacity);
    for (i = 0; i < capacity; i++)
        frame[i] = -1;
    for (i = 0; i < n; i++) {
        pagefound = 0;
        for (j = 0; j < capacity; j++)
            if (frame[j] == page[i]) {
                pagefound = 1;
                break;
            }
        if (!pagefound) {
            if (frameavailable < capacity) {
                frame[frameavailable] = page[i];
                frameavailable++;
            } else {
                frame[frameindex] = page[i];
                frameindex = (frameindex + 1) % capacity;
            }
            pagefault++;
        }
        printf("RS: %d |", page[i]);
        for (j = 0; j < capacity; j++) {
            if (frame[j] == -1)
                printf(" _");
            else
                printf(" %d", frame[j]);
        }
        printf("\n");
    }
    printf("Page faults: %d\n", pagefault);
    printf("Page Hits: %d\n", n-pagefault);
    printf("Page Fault Ratio: %d\n", pagefault*100/n);
    printf("Page Hit Ratio: %d", (n-pagefault)*100/n);
    return 0;
}
```

```c
//page_replacement_LRU
#include <stdio.h>
int main() {
    int pages[100], n, capacity, frame[10], page_faults = 0, frame_index = 0, page_age[10];
    printf("Enter the number of pages: ");
    scanf("%d", &n);
    printf("Enter the reference string (RS): ");
    for (int i = 0; i < n; i++)
        scanf("%d", &pages[i]);
    printf("Enter the capacity of frames: ");
    scanf("%d", &capacity);
    for (int i = 0; i < capacity; i++) {
        frame[i] = -1;
        page_age[i] = 0;
    }
    for (int i = 0; i < n; i++) {
        int page_found = 0;
        for (int j = 0; j < capacity; j++)
            if (frame[j] == pages[i]) {
                page_found = 1;
                page_age[j] = i + 1;
                break;
            }
        if (!page_found) {
            int lru_index = 0, min_age = page_age[0];
            for (int j = 1; j < capacity; j++) {
                if (page_age[j] < min_age) {
                    lru_index = j;
                    min_age = page_age[j];
                }
            }
            frame[lru_index] = pages[i];
            page_age[lru_index] = i + 1;
            page_faults++;
        }
        printf("RS: %d |: ", pages[i]);
        for (int j = 0; j < capacity; j++) {
            if (frame[j] == -1)
                printf(" _");
            else
                printf(" %d", frame[j]);
        }
        printf("\n");
    }
    printf("Page faults: %d\n", page_faults);
    printf("Page Hits: %d\n", n-page_faults);
    printf("Page Fault Ratio: %d\n", page_faults*100/n);
    printf("Page Hit Ratio: %d", (n-page_faults)*100/n);
    return 0;
}
```

```c
//page_replacement_LFU
#include <stdio.h>
int main() {
	int total_frames, total_pages, hit = 0;
	int m, n, page, flag, k, minimum_time, temp;
	int pages[20], frames[10], arr[20], time[20];
	printf("Enter the number of pages: ");
	scanf("%d", &total_pages);
	printf("Enter the capacity of frames: ");
	scanf("%d", &total_frames);
	for (m = 0;m < total_frames; m++)
		frames[m] = -1;
	for (m = 0; m < 25; m++)
		arr[m] = -1;
	printf("Enter the reference String (RS): ");
	for (m = 0; m < total_pages; m++)
		scanf("%d", &pages[m]);
	for (m = 0; m < total_pages; m++) {
		arr[pages[m]]++;
		time[pages[m]] = m;
		flag = 0;
		k = frames[0];
		for (n = 0; n < total_frames; n++) {
			if (frames[n] == -1 || frames[n] == pages[m]) {
				if (frames[n] != -1)
					hit++;
				flag = 1;
				frames[n] = pages[m];
				break;
			}
			if (arr[k] > arr[frames[n]])
				k = frames[n];
		}
		if (!flag) {
			minimum_time = 25;
			for (n = 0; n < total_frames; n++)
				if (arr[frames[n]] == arr[k] && time[frames[n]] < minimum_time) {
					temp=n;
					minimum_time = time[frames[n]];
				}
			arr[frames[temp]] = 0;
			frames[temp] = pages[m];
		}
		printf("RS: %d | ", pages[m]);
		for (int j = 0; j < total_frames; j++) {
			if (frames[j] == -1)
				printf(" _");
			else
				printf(" %d", frames[j]);
		}
		printf("\n");
	}
	printf("Page fault: %d \n", total_pages - hit);
	printf("Page hit: %d\n", hit);
	printf("Page fault ratio: %d\n", (total_pages - hit)*100/total_pages);
	printf("Page hit ratio: %d\n", hit*100/total_pages);
	return 0;
}
```

```c
//disk_fcfs
#include <stdio.h>
int main() {
    int ioq[20], i, n, ihead, tot;
    float seek = 0, avgs;
    printf("Enter the number of requests: ");
    scanf("%d", &n);
    printf("Enter the initial head position: ");
    scanf("%d", &ihead);
    ioq[0] = ihead;
    ioq[n + 1] = 0;
    printf("Enter the I/O queue requests:");
    for (i = 1; i <= n; i++)
        scanf("%d", &ioq[i]);
    ioq[n + 1] = ioq[n];
    printf("Order of requests served:");
    for (i = 0; i <= n; i++) {
        tot = ioq[i + 1] - ioq[i];
        if (tot < 0)
            tot = tot * -1;
        seek += tot;
        printf("%d --> ", ioq[i]);
    }
    avgs = seek / n;
    printf("\nTotal Seek time:\t%.2f\n", seek);
    printf("Average seek time:\t%.2f\n", avgs);
    return 0;
}
```

```c
//disk_scan
#include <stdio.h>
int main() {
    int ioq[20], i, n, j, ihead, temp, scan, tot;
    float seek = 0, avgs;
    printf("Enter the number of requests: ");
    scanf("%d", &n);
    printf("Enter the initial head position: ");
    scanf("%d", &ihead);
    ioq[0] = ihead;
    ioq[1] = 0;
    n += 2;
    printf("Enter the I/O queue requests: ");
    for (i = 2; i < n; i++)
        scanf("%d", &ioq[i]);
    for (i = 0; i < n - 1; i++)
        for (j = 0; j < n - 1; j++)
            if (ioq[j] > ioq[j + 1]) {
                temp = ioq[j];
                ioq[j] = ioq[j + 1];
                ioq[j + 1] = temp;
            }
    ioq[n] = ioq[n - 1];
    for (i = 0; i < n; i++)
        if (ihead == ioq[i]) {
            scan = i;
            break;
        }
    printf("Order of requests served: ");
    tot = 0;
    for (i = scan; i >= 0; i--) {
        tot = ioq[i] - ioq[i - 1];
        if (i == 0)
            tot = ioq[i] - ioq[scan + 1];
        if (tot < 0)
            tot = tot * -1;
        printf("%d--> ", ioq[i]);
    }
    for (i = scan + 1; i < n; i++) {
        tot = ioq[i + 1] - ioq[i];
        if (tot < 0)
            tot = tot * -1;
        printf("%d--> ", ioq[i]);
    }
    seek = ioq[scan] + ioq[n - 1];
    avgs = seek / (n - 2);
    printf("\nTotal Seek time:\t%.2f\n", seek);
    printf("Average seek time:\t%.2f\n", avgs);
    return 0;
}
```

```c
//disk_cscan
#include <stdio.h>
void main() {
    int ioq[20], i, n, j, ihead, itail, temp, scan, tot = 0;
    float seek = 0, avgs;
    printf("Enter the number of requests: ");
    scanf("%d", &n);
    ioq[0] = 0;
    printf("Enter the initial head position: ");
    scanf("%d", &ihead);
    ioq[1] = ihead;
    printf("Enter the maximum track limit: ");
    scanf("%d", &itail);
    ioq[2] = itail;
    n += 3;
    printf("Enter the I/O queue requests: ");
    for (i = 3; i < n; i++)
        scanf("%d", &ioq[i]);
    for (i = 0; i < n - 1; i++)
        for (j = 0; j < n - 1; j++)
            if (ioq[j] > ioq[j + 1]) {
                temp = ioq[j];
                ioq[j] = ioq[j + 1];
                ioq[j + 1] = temp;
            }
    for (i = 0; i < n + 1; i++)
        if (ihead == ioq[i]) {
            scan = i;
            break;
        }
    i = scan;
    temp = n;
    printf("Order of request served: ");
    while (i != temp) {
        if (i < temp - 1) {
            tot = ioq[i + 1] - ioq[i];
            if (tot < 0)
                tot = tot * -1;
            seek += tot;
        }
        printf("%d --> ", ioq[i]);
        i++;
        if (i == n) {
            i = 0;
            temp = scan;
            seek += itail;
        }
    }
    avgs = seek / (n - 3);
    printf("\nTotal Seek time\t\t: %.2f", seek);
    printf("\nAverage seek time\t: %.2f", avgs);
}
```